jc644 U.S. PTO

01/21/00

01-24-00

A

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

jc530 U.S. PTO 09/489878 01/21/00

In re Patent Application of: Manuvir Das

Title: METHODS FOR ENHANCING POINTER ANALYSES

Attorney Docket No.: 777.361US1

PATENT APPLICATION TRANSMITTAL

BOX PATENT APPLICATION
Assistant Commissioner for Patents
Washington, D.C. 20231

We are transmitting herewith the following attached items and information (as indicated with an "X"):

X Utility Patent Application under 37 CFR § 1.53(b) comprising:
- X Specification (38 pgs, including claims numbered 1 through 56 and a 1 page Abstract).
- X Formal Drawing(s) (11 sheets).
- X Signed Combined Declaration and Power of Attorney (3 pgs).
- X Check in the amount of $2,196.00 to pay the filing fee.

X Assignment of the invention to Microsoft Corporation (2 pgs) and Recordation Form Cover Sheet.
X Check in the amount of $40.00 to pay the Assignment recording fee.
X Return postcard.

The filing fee has been calculated below as follows:

	No. Filed	No. Extra	Rate	Fee
TOTAL CLAIMS	56 - 20 =	36	x 18 =	$648.00
INDEPENDENT CLAIMS	14 - 3 =	11	x 78 =	$858.00
[ ] MULTIPLE DEPENDENT CLAIMS PRESENTED				$0.00
BASIC FEE				$690.00
TOTAL				$2,196.00

Please charge any additional required fees or credit overpayment to Deposit Account No. 19-0743.

SCHWEGMAN, LUNDBERG, WOESSNER & KLUTH, P.A.
P.O. Box 2938, Minneapolis, MN 55402 (612-373-6900)

By: [signature]
Atty: D. C. Peter Chu
Reg. No. 41,676

Customer Number 21186

"Express Mail" mailing label number: EL510314740US   Date of Deposit: January 21, 2000

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Assistant Commissioner for Patents, Box Patent Application, Washington, D.C. 20231.

By: Shawn L. Hise   Signature: [signature]

UNITED STATES PATENT APPLICATION

METHODS FOR ENHANCING POINTER ANALYSES

INVENTOR

Manuvir Das

Schwegman, Lundberg, Woessner, & Kluth, P.A.
1600 TCF Tower
121 South Eighth Street
Minneapolis, Minnesota 55402

ATTORNEY DOCKET 777.361US1

MICROSOFT 144023.1

METHODS FOR ENHANCING POINTER ANALYSES

Technical Field

The technical field relates generally to program analyses. More particularly, it pertains to the analysis of pointers in programs.

Copyright Notice - Permission

Background

A program is a list of statements. This list of statements may be translated, through processes that include compilation, to produce an executable file that can cause a computer to perform a desired action. One type of statement is an assignment statement. An illustrative example of an assignment statement is x=y. This statement may be translated to mean that y is assigned to x, or more specifically, the value of the variable y is assigned to the variable x.

One type of variable is a pointer. Pointers are often used in programs because they offer flexibility in creating compact and efficient executable files. A pointer contains a location (or address) of another variable. Thus, a pointer points to another variable. Through a pointer, the value of another variable may be changed. In this way, a pointer indirectly references another variable.

It is beneficial to analyze programs in order to obtain information that may be used to improve them. In order to analyze a program that uses pointers, an

analysis is performed that focuses on statements that involve pointers. Such pointer analysis yields sets of information about pointers in the program. The precision of a pointer analysis is determined by the size of these sets of information. The larger the set the less precise is the information.

Current pointer analyses suffer from the extremes of an inverse relationship between time and information. One type of analysis can be performed quickly by using a technique of unification but provides imprecise results due to the production of large sets of information. Another analysis by Lars Ole Andersen provides results that are much more precise by producing small sets of information but requires a prohibitively long amount of time. See Lars Ole Andersen, Program Analysis and Specialization for the C Programing Language (1994) (published Ph.D. dissertation, University of Copenhagen). Thus, current pointer analyses are either too costly in terms of time or too imprecise in terms of information. Tools that rely on such pointer analyses such as optimizer and debugging tools have been constrained by having to make inferior assumptions about behaviors of programs. As the size of programs has increased with each generation of technology, such inferior assumptions may slow the improvement of programs and lead to the eventual lack of acceptance of such programs in the marketplace.

Thus, what is needed are systems, methods, and structures to enhance pointer analysis of programs so as to allow a desired level of analytical precision within a desired duration of analysis.

Summary

Systems, methods, and structures to support enhanced pointer analyses are described. An illustrative aspect includes a system for enhancing pointer analysis of a program. The program includes at least one source file. The system comprises a compiler to compile at least one source file to produce an intermediate language. The system further comprises a builder receptive to the intermediate language to build a tree that represents the source file. The system further comprises an analyzer

to analyze the tree to produce an object file. The object file contains at least one relationship between two variables in an assignment statement in the program. The relationship defines that a set of symbols relating to one of the two variables is a subset of a set of symbols relating to the other of the two variables.

Another illustrative aspect includes a method of analyzing pointers in a program. The method includes processing an assignment statement of two variables, forming a relationship such as a label relationship between two locations related to the two variables, and enforcing the relationship. The duration of the acts of processing an assignment statement and forming a relationship are about linearly proportional to the size of the program in theory and in practice. The method includes delaying the act of enforcing the relationship to enable the method to process each assignment statement in the program. The act of enforcing the relationship includes moving label information to create the label relationship. In one embodiment, such act of enforcing is about quadratically proportional to the size of the program in theory and is about linearly proportional to the size of the program in practice. Factoring, sharing, and other suitable techniques can be used such that the act of enforcing is about linearly proportional to the size of the program.

Another illustrative aspect includes a method of analyzing pointers in a program. The method comprises forming a location for at least one variable in the program. The location includes a label and a content. The method further comprises forming a relationship between two locations upon an assignment of a first variable and a second variable in the program. The relationship defines that a label of one of the two locations is a subset of a label of the other of the two locations. The contents of the two locations are selectively unified. In one aspect the second variable is assigned to the first variable.

Another illustrative aspect includes a method of analyzing pointers in a program. The method comprises forming a location for at least one variable in the program. The location includes a label and a content. The method further comprises forming a relationship between two locations upon an assignment of a first variable

and an address of a second variable in the program. The relationship defines that a label of one of the two locations is a subset of a label of the other of the two locations. The contents of the two locations are selectively unified. In one aspect, the address of the second variable is assigned to the first variable.

Another illustrative aspect includes a method of analyzing pointers in a program. The method comprises forming a location for at least one variable in the program. The location includes a label and a content. The method further comprises forming a relationship between two locations upon an assignment of a first variable and a dereference of a second variable in the program. The relationship defines that a label of one of the two locations is a subset of a label of the other of the two locations. The contents of the two locations are selectively unified. In one aspect, the dereference of the second variable is assigned to the first variable.

Another illustrative aspect includes a method of analyzing pointers in a program. The method comprises forming a location for at least one variable in the program. The location includes a label and a content. The method further comprises forming a relationship between two locations upon an assignment of a dereference of a first variable and a second variable in the program. The relationship defines that a label of one of the two locations is a subset of a label of the other of the two locations. The contents of the two locations are selectively unified. In one aspect, the second variable is assigned to the dereference of the first variable.

Another illustrative aspect includes a data structure to enhance pointer analysis in a program. The program includes at least one assignment statement of variables. The variable includes a name and a content. The data structure comprises a data member location and a data member flow to represent at least one label relationship. The data member location includes a data member label that includes at least one data member symbol, and a data member content that represents a content of the variable. The data member flow stores an address of another instantiation of the data structure if an assignment statement is defined for two variables, and the another instantiation of the data structure is related to one of the

two variables.

Brief Description of the Drawings

Figure 1 is a block diagram of a system according to one aspect of the present invention.

Figures 2A-2C illustrate a block diagram of a graph according to one aspect of the present invention.

Figure 3 is a process diagram of a method according to one aspect of the present invention.

Figures 4A-4C illustrate a block diagram of a graph according to one aspect of the present invention.

Figure 5 is a process diagram of a method according to one aspect of the present invention.

Figures 6A-6C illustrate a block diagram of a graph according to one aspect of the present invention.

Figure 7 is a process diagram of a method according to one aspect of the present invention.

Figures 8A-8C illustrate a block diagram of a graph according to one aspect of the present invention.

Figure 9 is a process diagram of a method according to one aspect of the present invention.

Figure 10 is a structure diagram of a data structure according to one aspect of the present invention.

Figure 11 is a block diagram of a system according to one aspect of the present invention.

Detailed Description

In the following detailed description of exemplary embodiments of the invention, reference is made to the accompanying drawings which form a part

hereof, and in which is shown, by way of illustration, specific exemplary embodiments in which the invention may be practiced. In the drawings, like numerals describe substantially similar components throughout the several views. These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention. Other embodiments may be utilized and structural, logical, electrical, and other changes may be made without departing from the spirit or scope of the present invention. The following detailed description is, therefore, not to be taken in a limiting sense, and the scope of the present invention is defined only by the appended claims.

Figure 1 is a block diagram of a system according to one aspect of the present invention. Figure 1 provides a brief, general description of a suitable computing environment in which the invention may be implemented. The invention will hereinafter be described in the general context of computer-executable program modules containing instructions executed by a personal computer (PC). Program modules include routines, programs, objects, components, data structures, etc., that perform particular tasks or implement particular abstract data types. Those skilled in the art will appreciate that the invention may be practiced with other computer-system configurations, including hand-held devices, multiprocessor systems, microprocessor-based programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like which have multimedia capabilities. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

Figure 1 shows a general-purpose computing device in the form of a conventional personal computer 120, which includes processing unit 121, system memory 122, and system bus 123 that couples the system memory and other system components to processing unit 121. System bus 123 may be any of several types, including a memory bus or memory controller, a peripheral bus, and a local bus, and

may use any of a variety of bus structures. System memory 122 includes read-only memory (ROM) 124 and random-access memory (RAM) 125. A basic input/output system (BIOS) 126, stored in ROM 124, contains the basic routines that transfer information between components of personal computer 120. BIOS 126 also contains start-up routines for the system. Personal computer 120 further includes hard disk drive 127 for reading from and writing to a hard disk (not shown), magnetic disk drive 128 for reading from and writing to a removable magnetic disk 129, and optical disk drive 130 for reading from and writing to a removable optical disk 131 such as a CD-ROM or other optical medium. Hard disk drive 127, magnetic disk drive 128, and optical disk drive 130 are connected to system bus 123 by a hard-disk drive interface 132, a magnetic-disk drive interface 133, and an optical-drive interface 134, respectively. The drives and their associated computer-readable media provide nonvolatile storage of computer-readable instructions, data structures, program modules, and other data for personal computer 120. Although the exemplary environment described herein employs a hard disk, a removable magnetic disk 129 and a removable optical disk 131, those skilled in the art will appreciate that other types of computer-readable media which can store data accessible by a computer may also be used in the exemplary operating environment. Such media may include magnetic cassettes, flash-memory cards, digital versatile disks, Bernoulli cartridges, RAMs, ROMs, and the like.

Program modules may be stored on the hard disk, magnetic disk 129, optical disk 131, ROM 124 and RAM 125. Program modules may include operating system 135, one or more application programs 136, other program modules 137, and program data 138. A user may enter commands and information into personal computer 120 through input devices such as a keyboard 140 and a pointing device 142. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 121 through a serial-port interface 146 coupled to system bus 123; but they may be connected through other interfaces not shown in

Figure 1, such as a parallel port, a game port, or a universal serial bus (USB). A monitor 147 or other display device also connects to system bus 123 via an interface such as a video adapter 148. In addition to the monitor, personal computers typically include other peripheral output devices such as a sound adapter 156, speakers 157, and further devices such as printers.

Personal computer 120 may operate in a networked environment using logical connections to one or more remote computers such as remote computer 149. Remote computer 149 may be another personal computer, a server, a router, a network PC, a peer device, or other common network node. It typically includes many or all of the components described above in connection with personal computer 120; however, only a storage device 150 is illustrated in Figure 1. The logical connections depicted in Figure 1 include local-area network (LAN) 151 and a wide-area network (WAN) 152. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

When placed in a LAN networking environment, PC 120 connects to local network 151 through a network interface or adapter 153. When used in a WAN networking environment such as the Internet, PC 120 typically includes modem 154 or other means for establishing communications over network 152. Modem 154 may be internal or external to PC 120, and connects to system bus 123 via serial-port interface 146. In a networked environment, program modules, such as those comprising Microsoft® Word which are depicted as residing within PC 120 or portions thereof may be stored in remote storage device 150. Of course, the network connections shown are illustrative, and other means of establishing a communications link between the computers may be substituted.

Software may be designed using many different methods, including object-oriented programming methods. C++ is one example of common object-oriented computer programming languages that provides the functionality associated with object-oriented programming. Object-oriented programming methods provide a means to encapsulate data members (variables) and member functions (methods)

that operate on that data into a single entity called a class. Object-oriented programming methods also provide a means to create new classes based on existing classes.

An object is an instance of a class. The data members of an object are attributes that are stored inside the computer memory, and the methods are executable computer code that act upon this data, along with potentially providing other services. The notion of an object is exploited in the present invention in that certain aspects of the invention are implemented as objects in one embodiment.

An interface is a group of related functions that are organized into a named unit. Each interface may be uniquely identified by some identifier. Interfaces have no instantiation, that is, an interface is a definition only without the executable code needed to implement the methods which are specified by the interface. An object may support an interface by providing executable code for the methods specified by the interface. The executable code supplied by the object must comply with the definitions specified by the interface. The object may also provide additional methods. Those skilled in the art will recognize that interfaces are not limited to use in or by an object-oriented programming environment.

The embodiments of the present invention focus on enhancing pointer analyses. As mentioned hereinbefore, a program is a list of statements. Depending on the programming language, these statements can be especially expressive and may be classified into many different types. One type includes an assignment of a complicated expression such as "x=y+z*2". The embodiments of the present invention simplify these different types into four so as to ease the process of pointer analysis. These four types are discussed in more detail below.

Now, for illustrative purposes only, suppose one of the simplified four types of assignment statement is defined for the variables x and y in the program. Such an assignment statement causes the embodiments of the present invention to create a relationship between a location related to the variable x and a location related to the variable y. Without this relationship, a pointer analysis may be constrained by the

extremes of the inverse relationship between time and information. This relationship allows a pointer analysis to selectively retain information for a desired analytical precision within a desired duration of analysis.

The terms "pointer" or "pointer type," hereinbefore and hereinafter, are understood to mean the inclusion of a predefined data type in a programming language. However, these terms include the type conversion that may occur automatically to variables in a program, or type casting that may occur by forcing variables in a program to hold values of a given type.

Figures 2A-2C illustrate a block diagram of a graph according to one aspect of the present invention. A number of nodes appear in the graph of Figures 2A-2C. A node graphically represents a location. The location represents a variable in a program in one embodiment. In another embodiment, the location is related to a variable through at least one pointer. The location includes a label and a content. The label contains at least one symbol. The term "symbol" is understood to mean the inclusion of a name or an identifier of a variable. The content contains a value. For illustrative purposes, suppose that a location A represents a pointer variable. Then, the content of the location A contains an address of another location, and for the sake of the illustration, this other location is a location B. A line graphically emanates from the content area of the node that represents the location A and graphically points to another node that represents the location B. The location B is also called the pointed-to location of the location A.

Figure 2A shows a graph following the next sequence of processing. A graph 200 shows pointer relationships between various nodes before an assignment statement of interest is defined in a program. The graph 200 includes a node 202 that represents a variable x. The node 202 includes a label 202_A and a content 202_B. The label 202_A contains a symbol x. A line 202_C shows that there is a pointer relationship between node 202 and node 204. Therefore, the node 202 represents a pointer variable x in the program, and the node 204 represents a pointed-to location of the variable x. In one embodiment, only one line can emanate from any single node to

represent a pointer relationship with another node. A pointer relationship also exists between nodes 204 and 206 through a line 204_C. In one embodiment, the node 204 represents a level of indirection, and the node 206, which is a pointed-to location of the node 204, represents another level. A line 206_C shows that there may be other pointer relationships related to the node 206.

The graph 200 also includes a node 208 that represents a variable y. The node 208 includes a label 208_A and a content 208_B. The label 208_A contains a symbol y. A line 208_C shows a pointer relationship between nodes 208 and 210. Therefore, the node 208 represents a pointer variable y in the program, and the node 210 is a pointed-to location of the variable y. A line 210_C shows a pointer relationship between nodes 210 and 212. In one embodiment, the node 210 represents a level of indirection, and the node 212, which is a pointed-to location of the node 210, represents another level. A line 212_C shows a pointer relationship between nodes 212 and others (not shown).

Hereinafter, for clarity purposes, many of the reference numbers are eliminated from subsequent drawings so as to focus on the portion of interest of the graphs of the various figures.

Figure 2B shows a graph following the next sequence of processing. For illustrative purposes, suppose the assignment statement defines that "x=y" in the program. In one embodiment, such an assignment statement creates the relationship between a pointed-to location of the variable x and a pointed-to location of the variable y. In one embodiment, the relationship defines that the label of the pointed-to location of the variable y is a subset of the label of the pointed-to location of the variable x. This subset is the information that can be selectively retained to achieve the desired analytical precision.

The graph 200 represents this relationship through a line 201. In one embodiment, the line 201 emanates from the node 210 to point to the node 204. In one embodiment, the line 201 is distinguished from other lines in the graph 200 by having an "f" appear above the line 201. The line 201 may be referred to as a flow

line. In one embodiment, at least one flow line may emanate from any single node to show a label relationship. The direction of the line as shown by the arrowhead indicates that the label of the node 210 is a subset of the label of the node 204. In one embodiment, since the node 204 and the node 210 are in the same level of indirection, the line 201 defines a label relationship that is at the same level of indirection.

The assignment statement may cause a selective unification. The term "selective unification" means the unification of information, and whether such unification will take place is based on a decision by the user or the program. In one embodiment, the content of the pointed-to location of the variable x is selectively unified with the content of the pointed-to location of the variable y. The graph 200 represents this unification by including a marquee 203 around the node 206 and the node 212. The process of unification is discussed by Bjarne Steensgaard, <u>Points-to Analysis In Almost Linear Time</u>, Conference Record of the Twenty-Third ACM Symposium on Principles of Programming Languages, p. 32-41 (January 1996). Such process of unification does not limit the embodiments of the present invention, and as such, will not be presented here in full.

Figure 2C shows a graph following the next sequence of processing. The graph 200, after the process of unification, includes a node 205. The node 205 represents the unification of the nodes 206 and 212. The content of the pointed-to location of the variable x, which is represented by the node 206, and the content of the pointed-to location of the variable y, which is represented by the node 212, contains the address of the location represented by the node 205. Thus, both lines 204_c and 210_c point to the node 205.

Figure 3 is a process diagram of a method according to one aspect of the present invention. A method 300 includes an act 302 for forming a location. The location includes a label and a content. The method 300 includes an act 304 for forming a relationship between two locations upon an assignment of two variables in a program. For illustrative purposes only, suppose that the assignment defines

"x=y." The act 304 includes an act 306 for defining that a label of one of the two locations is a subset of a label of the other of the two locations. If the variables x and y are pointer variables, then the act 306 defines that the label of the pointed-to location of the variable y is a subset of the label of the pointed-to location of the variable x. The method 300 also includes an act 308 for selective unification of the contents of the two locations.

In another embodiment, the method 300 may be considered as a process for determining whether a program is well typed or correctly typed under a pointer analysis. This process uses a combination of set theory, sentential calculus, predicate calculus, and metalogic to express such determination. The domain of the determination includes:

$s \in$ Symbols

$\tau \in$ Locations $::= (\varphi, \alpha)$

$\varphi \in$ Labels $::= \{s_1, \ldots, s_n\}$

$\alpha \in$ Values $::= \perp \mid ptr(\tau)$

Thus, s is an element of symbols, and the term "symbol" has been discussed hereinbefore. τ is an element of locations, and the term "location" has been discussed hereinbefore, but in this instance, the term "location" is expressed as including φ and α. φ is an element of labels, and the term "label" has been discussed hereinbefore, but in this instance, the term "label" is further expressed as a set of symbols. α is an element of values, and the term "value" is understood to mean the inclusion of "$\perp$" or ptr(τ). The term "$\perp$" is used in metalogic to mean falsehood, but in this instance, the term "$\perp$" means the inclusion of an initial value or a value that is not a pointer. The term ptr(τ), in predicate calculus, means the inclusion a pointed-to location of τ where τ is a location of a pointer variable, and therefore, α may contain an address of another location.

The following relational logic expression defines the conditions for a valid less-than-or-equal-to relationship in determining whether the program is well typed or correctly typed:

$$\mathrm{ptr}(\varphi, \alpha) \leq \mathrm{ptr}(\varphi', \alpha) \leftrightarrow \varphi \subseteq \varphi'$$

For illustrative purposes only, the term "ptr(φ, α)" means a pointed-to location that has φ and α. The term "ptr(φ', α)" means a pointed-to location that has φ' and α. The logic expression includes the following meaning: Two pointed-to locations would satisfy the relational expression if and only if the φ of one pointed-to location is a subset of the φ' of the other pointed-to location, and that each pointed-to location's α is unified.

The determination of whether a program is well typed or correctly typed under a pointer analysis for the assignment statement "x=y" includes the following type inference rule:

$$\frac{A \vdash x : (\varphi, \alpha) \qquad A \vdash y : (\varphi', \alpha') \qquad \alpha' \leq \alpha}{A \vdash \mathrm{welltyped}(x=y)}$$

This type inference rule includes the following meaning. The expression "A ⊢ well typed(x=y)" indicates that given all the knowledge one has so far (A), a program is well typed for the assignment statement "x=y" if all the expressions above the bar hold true. First, x is a variable that is associated with φ and α. Second, y is a variable that is associated with φ' and α'. And third, α' must be less than or equal to α. In order for α' and α to be in condition for a valid less-than-or-equal-to relationship, the relational logic expression discussed hereinbefore is applied to the statement α' ≤ α. Accordingly, since α' is associated with the variable y and it is on the left side of the symbol "≤", the pointed-to location of the variable y is compared to the pointed-to location of the variable x. Therefore, the φ

of the pointed-to location of y is adapted to be a subset of the φ of the pointed-to location of x, and the α of the pointed-to location of y is adapted to be unified with the α of the pointed-to location of x.

Figures 4A-4C illustrate a block diagram of a graph according to one aspect of the present invention. Figures 4A-4C contain elements similar to those discussed in Figures 2A-2C. These elements appear in Figures 4A-4C with the last two digits of the numerical nomenclature matching those in Figures 2A-2C. For clarity purposes, the hereinbefore discussion related to these elements is incorporated here in full.

Figure 4A shows a graph following the next sequence of processing. The graph 400 includes elements similar to those in Figure 2A. The graph 400 also includes a ghost of a node 414 whose content includes the address of the variable y. The purpose of the node 414 is to aid the discussion to follow.

Figure 4B shows a graph following the next sequence of processing. For illustrative purposes only, suppose an assignment statement defines "x=&y" in the program. The symbol "&" is understood to mean the inclusion of a unary operator in a programming language to obtain an address of a variable. Thus, for illustrative purposes only, the expression "&y" can be thought to be equivalent to a pointer to the variable y since this pointer would contain an address of the variable y. The pointer is illustratively shown as node 414. In one embodiment, such an assignment statement creates a relationship between a pointed-to location of the variable x and the variable y. In one embodiment, the relationship defines that the label of the location of the variable y is a subset of the label of the pointed-to location of the variable x. This subset is the information that can be selectively retained to achieve the desired analytical precision.

A line 401 shows the relationship between the node 408 and the node 404. The direction of the line 401 as shown by the arrowhead indicates that the label of the node 408 is a subset of the label of the node 404. In one embodiment, since the

node 404 and the node 408 are in different levels of indirection, the line 401 defines a label relationship that is at different levels of indirection. The marquee 403 shows that the selective unification occurs between nodes 410 and 406.

Figure 4C shows a graph following the next sequence of processing. The graph 400, after the process of unification, shows a node 405. The node 405 appears as a pointed-to location for the nodes 408 and 404.

Figure 5 is a process diagram of a method according to one aspect of the present invention. Figure 5 contains acts similar to those discussed in Figure 3. These acts appear in Figure 5 with the last two digits of the numerical nomenclature matching those in Figure 3. For clarity purposes, the hereinbefore discussion related to these acts is incorporated here in full.

For illustrative purposes only, suppose that the assignment defines “x=&y”. The act 504 includes an act 506 for defining that a label of one of the two locations is a subset of a label of the other of the two locations. If x is a pointer variable and y is a variable, then the act 506 defines that the label of the location of the variable y is a subset of the label of the pointed-to location of the variable x.

In another embodiment, the method 500 may be considered as a process for determining whether a program is well typed or correctly typed under a pointer analysis. The domain of the determination is similar to those discussed hereinbefore in Figure 3, and that domain is incorporated here in full. The hereinbefore discussion of the relational logic expression for defining the conditions for a valid less-than-or-equal-to relationship is also incorporated here in full.

The determination of whether a program is well typed or correctly typed under a pointer analysis for the assignment statement “x=&y” includes the following type inference rule:

$$\begin{array}{l} A \vdash x : (\varphi, \alpha) \\ A \vdash y : \tau \\ ptr(\tau) \leq \alpha \\ \hline A \vdash welltyped(x{=}\&y) \end{array}$$

This type inference rule includes the following meaning. The expression "A $\vdash$ well typed(x=&y)" indicates that given all the knowledge one has so far (A), a program is well typed for the assignment statement "x=&y" if all the expressions above the bar hold true. First, x is a variable that is associated with φ and α. Second, y is a variable that is associated with τ. And third, ptr(τ) must be less than or equal to α; in other words, the pointer to a location of τ must be less than or equal to α. In order for ptr(τ) and α to be in condition for a valid less-than-or-equal-to relationship, the relational logic expression discussed hereinbefore is applied to the statement ptr(τ) $\leq \alpha$. Accordingly, since τ is a location associated with the variable y and since ptr(τ) has already satisfied the relational logic expression on the left side of the symbol "$\leq$", the location of the variable y is compared to the pointed-to location of the variable x. Therefore, the φ of the location of the variable y is adapted to be a subset of the φ of the pointed-to location of x, and the α of the location of the variable y is adapted to be unified with the α of the pointed-to location of x.

Figures 6A-6C illustrate a block diagram of a graph according to one aspect of the present invention. Figures 6A-6C contain elements similar to those discussed in Figures 2A-2C. These elements appear in Figures 6A-6C with the last two digits of the numerical nomenclature matching those in Figures 2A-2C. For clarity purposes, the hereinbefore discussion related to these elements is incorporated here in full.

Figure 6A shows a graph following the next sequence of processing. The graph 600 includes nodes 616 and 618. Node 616 is a pointed-to location of the node 606. Node 618 is a pointed-to location of the node 612.

Figure 6B shows a graph following the next sequence of processing. For illustrative purposes only, suppose an assignment statement defines "x=*y" in the program. The symbol "*" is understood to mean the inclusion of a unary operator in a programming language to dereference a pointer variable. Thus, for illustrative purposes only, the expression "*y" can be thought to be equivalent to a pointed-to location of the variable y. In one embodiment, such an assignment statement creates a relationship between a pointed-to location of the variable x and a pointed-to location of a pointed-to location of the variable y. In one embodiment, the relationship defines that the label of the pointed-to location of the pointed-to location of the variable y is a subset of the label of the pointed-to location of the variable x. This subset is the information that can be selectively retained to achieve the desired analytical precision.

A line 601 shows the relationship between the node 612 and the node 604. The direction of the line 601 as shown by the arrowhead indicates that the label of the node 612 is a subset of the label of the node 604. In one embodiment, since the node 612 and the node 604 are in different levels of indirection, the line 601 defines a label relationship that is at different levels of indirection. The marquee 603 shows that the selective unification occurs between nodes 618 and 606.

Figure 6C shows a graph following the next sequence of processing. The graph 600, after the process of unification, shows a node 605. The node 605 appears as a pointed-to location for the nodes 612 and 604.

Figure 7 is a process diagram of a method according to one aspect of the present invention. Figure 7 contains acts similar to those discussed in Figure 3. These acts appear in Figure 7 with the last two digits of the numerical nomenclature matching those in Figure 3. For clarity purposes, the hereinbefore discussion related to these acts is incorporated here in full.

For illustrative purposes only, suppose that the assignment defines "x=*y". The act 704 includes an act 706 for defining that a label of one of the two locations is a subset of a label of the other of the two locations. If x and y are pointer variables, then the act 706 defines that the label of the pointed-to location of the pointed-to location of the variable y is a subset of the label of the pointed-to location of the variable x.

In another embodiment, the method 700 may be considered as a process for determining whether a program is well typed or correctly typed under a pointer analysis. The domain of the determination is similar to those discussed hereinbefore in Figure 3, and that domain is incorporated here in full. The hereinbefore discussion of the relational logic expression for defining the conditions for a valid less-than-or-equal-to relationship is also incorporated here in full.

The determination of whether a program is well typed or correctly typed under a pointer analysis for the assignment statement "x=*y" includes the following type inference rule:

$$\frac{\begin{array}{l} A \vdash x : (\varphi, \alpha) \\ A \vdash y : (\varphi', ptr(\tau)) \\ \tau = (\varphi'', \alpha'') \\ \alpha'' \leq \alpha \end{array}}{A \vdash \text{welltyped}(x=*y)}$$

This type inference rule includes the following meaning. The expression "A $\vdash$ well typed(x=*y)" indicates that given all the knowledge one has so far (A), a program is well typed for the assignment statement "x=*y" if all the expressions above the bar hold true. First, x is a variable that is associated with φ and α. Second, y is a variable that is associated with φ' and ptr(τ). Third, τ is a location with $\varphi''$and α''. Fourth, α'' must be less than or equal to α. In order for α'' and α to be in condition for a valid less-than-or-equal-to relationship, the relational logic expression discussed hereinbefore is applied to the statement $\alpha'' \leq \alpha$. Accordingly,

since α'' is associated with τ, since τ is a pointed-to location of the variable y, the pointed-to location of the pointed-to location of the variable y is compared with the pointed-to location of the variable x. Therefore, the φ of a pointed-to location of the pointed-to location of the variable y must be a subset of the φ of the pointed-to location of x, and the α of a pointed-to location of a pointed-to location of the variable y must be unified with the α of the pointed-to location of x.

Figures 8A-8C illustrate a block diagram of a graph according to one aspect of the present invention. Figures 8A-8C contain elements similar to those discussed in Figures 6A-6C. These elements appear in Figures 8A-8C with the last two digits of the numerical nomenclature matching those in Figures 6A-6C. For clarity purposes, the hereinbefore discussion related to these elements is incorporated here in full.

Figure 8A shows a graph following the next sequence of processing. The graph 800 includes nodes 816 and 818. Node 816 is a pointed-to location of the node 806. Node 818 is a pointed-to location of the node 812.

Figure 8B shows a graph following the next sequence of processing. For illustrative purposes only, suppose an assignment statement defines "*x=y" in the program. Thus, for illustrative purposes only, the expression "*x" can be thought to be equivalent to a pointed-to location of the variable x. In one embodiment, such an assignment statement creates a relationship between a pointed-to location of a pointed-to location of the variable x and a pointed-to location of the variable y. In one embodiment, the relationship defines that the label of the pointed-to location of the variable y is a subset of the label of the pointed-to location of the pointed-to location of the variable x. This subset is the information that can be selectively retained to achieve the desired analytical precision.

A line 801 shows the relationship between the node 810 and the node 806. The direction of the line 801 as shown by the arrowhead indicates that the label of the node 810 is a subset of the label of the node 806. In one embodiment, since the

node 810 and the node 806 are in different levels of indirection, the line 801 defines a label relationship that is at different levels of indirection. The marquee 803 shows that the selective unification occurs between nodes 812 and 816.

Figure 8C shows a graph following the next sequence of processing. The graph 800, after the process of unification, shows a node 805. The node 805 appears as a pointed-to location for the nodes 810 and 806.

Figure 9 is a process diagram of a method according to one aspect of the present invention. Figure 9 contains acts similar to those discussed in Figure 7. These acts appear in Figure 9 with the last two digits of the numerical nomenclature matching those in Figure 7. For clarity purposes, the hereinbefore discussion related to these acts is incorporated here in full.

For illustrative purposes only, suppose that the assignment defines "*x=y". The act 904 includes an act 906 for defining that a label of one of the two locations is a subset of a label of the other of the two locations. If x and y are pointer variables, then the act 906 defines that the label of the pointed-to location of the variable y is a subset of the label of the pointed-to location of the pointed-to location of the variable x.

In another embodiment, the method 900 may be considered as a process for determining whether a program is well typed or correctly typed under a pointer analysis. The domain of the determination is similar to those discussed hereinbefore in Figure 3, and that domain is incorporated here in full. The hereinbefore discussion of the relational logic expression for defining the conditions for a valid less-than-or-equal-to relationship is also incorporated here in full.

The determination of whether a program is well typed or correctly typed under a pointer analysis for the assignment statement "*x=y" includes the following type inference rule:

$$\frac{\begin{array}{l} A \vdash x : (\varphi', ptr(\tau)) \\ A \vdash y : (\varphi, \alpha) \\ \tau = (\varphi'', \alpha'') \\ \alpha \leq \alpha'' \end{array}}{A \vdash welltyped(*x=y)}$$

This type inference rule includes the following meaning. The expression "$A \vdash$ well typed(*x=y)" indicates that given all the knowledge one has so far (A), a program is well typed for the assignment statement "*x=y" if all the expressions above the bar hold true. First, x is a variable that is associated with φ' and ptr(τ). Second, y is a variable that is associated with φ and α. Third, τ is a location with φ'' and α''. Fourth, α must be less than or equal to α''. In order for α and α'' to be in condition for a valid less-than-or-equal-to relationship, the relational logic expression discussed hereinbefore is applied to the statement $\alpha \leq \alpha''$. Accordingly, since α is associated with the variable y, the pointed-to location of the variable y is compared with the pointed-to location of the pointed-to location of the variable x. Therefore, the φ of the pointed-to location of the variable y must be a subset of the φ of the pointed-to location of the pointed-to location of the variable x, and the α of a pointed-to location of the variable y must be unified with the α of the pointed-to location of the pointed-to location of the variable x.

In the discussion hereinbefore and hereinafter, the act of making a label of a location a subset of a label of another location includes an act of propagating the label from one location to another location such that the subset is formed. In one embodiment, that act of propagating is delayed for a predetermined period of time so as to allow the processing of each assignment statement in a program.

Figure 10 is a structure diagram of a data structure according to one aspect of the present invention. A data structure 1000 includes a data member location 1002. The data member location 1002 includes one data member label 1004. The data

member label 1004 includes at least one data member symbol that represents a name of a variable. The data member location 1002 also includes a data member content 1008. The data member content 1008 represents a content of a variable or a unification of at least two variables.

The data structure 1000 includes a data member flow 1012. The data member flow 1012 represents at least one label relationship between two instantiations of the data structure. In one embodiment, the data member flow 1012 stores an address of an instantiation of the data structure 1000 if an assignment statement is defined for two variables, and the instantiation is related to one of the two variables.

The data structure 1000 optionally includes a method member propagate 1014. In one embodiment, the method member propagate causes a propagation of at least one data member symbol 1006 so as to make the data member label 1004 of one instantiation of the data structure 1000 a subset of a data member label 1004 of another instantiation of the data structure 1000. The data structure 1000 also optionally includes a method member unify 1016. In one embodiment, the method member unify 1016 merges a data member label 1004 of one instantiation of the data structure 1000 with a data member label 1004 of another instantiation of the data structure 1000, and unifies a data member content 1008 of one instantiation of the data structure 1000 with a data member content 1008 of another instantiation of the data structure 1000.

Figure 11 is a block diagram of a system according to one aspect of the present invention. System 1100 includes a source file 1102. The source file 1102 contains a program or a portion of a program. In the embodiment where the source file 1102 contains only a portion of the program, other portions of the program are distributed in other source files (not shown).

System 1100 includes a compiler 1104. In one embodiment, the compiler 1104 includes any one of the compilers available in Visual Studio Suite, a product

of Microsoft Corporation. In a further embodiment, the compiler 1104 includes a C language compiler.

In system 1100, a source file 1102 that contains a program or portions of a program is input into the compiler 1104. The compiler 1104 translates the statements of the source file 1102 into an equivalent set of statements in a file 1106 that is in an intermediate language. The file 1106 is input into a builder 1108. The builder 1108 builds a tree 1110 that is a representation of the set of statements of file 1106. This tree 1110 contains grammatical phrases of statements in the file 1106. In one embodiment, this tree 1110 is an abstract syntax tree (hereinafter, AST).

The tree 1110 is then input into an analyzer 1112. The analyzer 1112 analyzes the tree 1110 and produces an object file 1114. The object file 1114 contains information for the source file 1102. In one embodiment, the object file 1114 contains at least one relationship between two variables in an assignment statement in the source file 1102.

The system 1100 includes a linker 1118. The object file 1114 and other object files that were generated previously such as object files 1114_0, 1114_1, 1114_2, …, and 1114_N are input into the linker 1118. The linker 1118 links the pointer information in each of the object files together and produces pointer information for object files 1114, 1114_0, 1114_1, 1114_2, …, and 1114_N. If the original source files of these object files constitute a program, then the linker 1118 produces information for the entire program. In one embodiment, the linker 1118 produces information for pointer analysis for the entire program.

Conclusion

Methods have been described to enhance pointer analysis for programs. Such enhancement allows tools such as program optimizers, error detection tools, and user feedback tools to make superior assumptions about programs under analysis. One result from such enhancement includes software products that may run faster, contain fewer bugs, or both. These methods allow a pointer analysis to

scale well to large programs while providing a desired level of analytical precision within a desired duration of analysis.

Although the specific embodiments have been illustrated and described herein, it will be appreciated by those of ordinary skill in the art that any arrangement which is calculated to achieve the same purpose may be substituted for the specific embodiments shown. This application is intended to cover any adaptations or variations of the present invention. It is to be understood that the above description is intended to be illustrative, and not restrictive. Combinations of the above embodiments and other embodiments will be apparent to those of skill in the art upon reviewing the above description. The scope of the invention includes any other applications in which the above structures and fabrication methods are used. Accordingly, the scope of the invention should only be determined with reference to the appended claims, along with the full scope of equivalents to which such claims are entitled.

I claim:

1. A method for enhancing pointer analysis, the method comprising:

processing an assignment between two variables in a program, wherein processing an assignment includes forming a relationship between two locations that are related to the two variables, wherein each location includes a label and a content, and wherein a content of one of the two locations is selectively unified with a content of an other of the two locations; and

propagating a label of the one of the two locations to a label of the other of the two locations such that the label of the one of the two locations is a subset of the other of the two locations.

2. The method of claim 1, wherein the act of propagating is delayed for a predetermined period of time so as to allow the act of processing an assignment to be executed for each assignment in the program.

3. The method of claim 1, further comprising forming a points-to graph by iterating the act of processing an assignment for each assignment in the program.

4. The method of claim 3, wherein forming a points-to graph includes forming a plurality of nodes, and forming a flow line between two nodes so as to represent the relationship between the two locations.

5. A computer readable medium having instructions stored thereon for causing a computer to perform a method for enhancing pointer analysis, the method comprising:

processing an assignment between two variables in a program, wherein processing an assignment includes forming a relationship between two locations that

are related to the two variables, wherein each location includes a label and a content, and wherein a content of one of the two locations is selectively unified with a content of an other of the two locations; and

propagating a label of the one of the two locations to the label of the other of the two locations such that the label of the one of the two locations is a subset of the other of the two locations.

6. A method of analyzing pointers in a program, the method comprising:

forming a location for at least one variable in the program, wherein the location includes a label and a content; and

defining a relationship between two locations upon an assignment in the program, wherein a label of one of the two locations is associated with a label of an other of the two locations, and wherein contents of the two locations are selectively unified.

7. The method of claim 6, further comprising propagating the label of the one of the two locations to the other of the two locations so as to make the label of the one of the two locations a subset of the label of the other of the two locations.

8. The method of claim 6, wherein forming a location includes forming a location that points to another location, and wherein the another location defines a pointed-to location of the location.

9. The method of claim 8, further comprising defining at least one level, wherein the at least one level is defined by at least one location, wherein a pointed-to location of the at least one location defines another level.

10. The method of claim 9, wherein defining a relationship includes defining a relationship between the two locations that are in the same level.

11. The method of claim 10, wherein defining a relationship includes defining a relationship between the two locations that are in different levels.

12. A method of analyzing pointers in a program, the method comprising:

forming a location for at least one variable in the program, wherein the location includes a label and a content; and

forming a relationship between two locations upon an assignment of a first variable and a second variable in the program, wherein the relationship defines that a label of one of the two locations is a subset of a label of an other of the two locations, and wherein contents of the two locations are selectively unified.

13. The method of claim 12, wherein forming a location includes forming a location that points to another location, and wherein the another location defines a pointed-to location of the location.

14. The method of claim 13, wherein forming a relationship includes forming a relationship between two locations upon an assignment of a first variable and a second variable, wherein the second variable is assigned to the first variable.

15. The method of claim 14, wherein forming a location includes forming a first location for the first variable and forming a second location for the second variable, wherein the first location points to the other of the two locations, and wherein the second location points to the one of the two locations.

16. The method of claim 15, further comprising determining that the program is correctly typed given that the second variable is assigned to the first variable if and only if a label of a pointed-to location of the second location is a subset of a label of a pointed-to location of the first location, and wherein a content of the pointed-to

location of the first location is selectively unified with a content of the pointed-to location of the second location.

17. A method of analyzing pointers in a program, the method comprising:

forming a location for at least one variable in the program, wherein the location includes a label and a content; and

forming a relationship between two locations upon an assignment of a first variable and an address of a second variable in the program, wherein the relationship defines that a label of one of the two locations is a subset of a label of an other of the two locations, and wherein contents of the two locations are selectively unified.

18. The method of claim 17, wherein forming a location includes forming a location that points to another location, and wherein the another location defines a pointed-to location of the location.

19. The method of claim 18, wherein forming a relationship includes forming a relationship between two locations upon an assignment of a first variable and an address of the second variable, wherein the address of the second variable is assigned to the first variable.

20. The method of claim 19, wherein forming a location includes forming a first location for the first variable and forming a second location for the second variable, wherein the first location points to the other of the two locations, and wherein the second location is the one of the two locations.

21. The method of claim 20, further comprising determining that the program is correctly typed given that the address of the second variable is assigned to the first variable if and only if a label of the second location is a subset of a label of a pointed-to location of the first location, and wherein a content of the pointed-to

location of the first location is selectively unified with a content of the second location.

22. A method of analyzing pointers in a program, the method comprising:

forming a location for at least one variable in the program, wherein the location includes a label and a content; and

forming a relationship between two locations upon an assignment of a first variable and a dereference of a second variable in the program, wherein the relationship defines that a label of one of the two locations is a subset of a label of an other of the two locations, and wherein contents of the two locations are selectively unified.

23. The method of claim 22, wherein forming a location includes forming a location that points to another location, and wherein the another location defines a pointed-to location of the location.

24. The method of claim 23, wherein forming a relationship includes forming a relationship between two locations upon an assignment of a first variable and a dereference of a second variable, wherein the dereference of the second variable is assigned to the first variable.

25. The method of claim 24, wherein forming a location includes forming a first location for the first variable and forming a second location for the second variable, wherein the first location points to the other of the two locations, wherein the second location points to a first pointed-to location, and wherein the first pointed-to location points to the one of the two locations to define a second pointed-to location.

26. The method of claim 25, further comprising determining that the program is correctly typed given that the dereference of the second variable is assigned to the

first variable if and only if a label of the second pointed-to location is a subset of a label of a pointed-to location of the first location, and wherein a content of the pointed-to location of the first location is selectively unified with a content of the second pointed-to location.

27. A method of analyzing pointers in a program, the method comprising:

forming a location for at least one variable in the program, wherein the location includes a label and a content; and

forming a relationship between two locations upon an assignment of a dereference of a first variable and a second variable in the program, wherein the relationship defines that a label of one of the two locations is a subset of a label of an other of the two locations, and wherein contents of the two locations are selectively unified.

28. The method of claim 27, wherein forming a location includes forming a location that points to another location, and wherein the another location defines a pointed-to location of the location.

29. The method of claim 28, wherein forming a relationship includes forming a relationship between two locations upon an assignment of a dereference of a first variable and a second variable, wherein the second variable is assigned to the dereference of the first variable.

30. The method of claim 29, wherein forming a location includes forming a first location for the first variable and forming a second location for the second variable, wherein the first location points to a pointed-to location that points to the other of the two locations to defined a first pointed-to location, wherein the second location points to the one of the two locations.

31. The method of claim 30, further comprising determining that the program is correctly typed given that the second variable is assigned to the dereference of the first variable if and only if a label of a pointed-to location of the second location is a subset of a label of the first pointed-to location, and wherein a content of the first pointed-to location is selectively unified with a content of the pointed-to location of the second location.

32. A computer readable medium having instructions stored thereon for causing a computer to perform a method of analyzing pointers in a program, the method comprising:

forming a location for at least one variable in the program, wherein the location includes a label and a content; and

defining a relationship between two locations upon an assignment in the program, wherein a label of one of the two locations is defined as a subset of a label of an other of the two locations, and wherein contents of the two locations are selectively unified.

33. The method of claim 32, wherein defining a relationship includes defining a relationship between two locations upon an assignment of a first variable and a second variable in the program, wherein the first variable and the second variable are pointers.

34. The method of claim 32, wherein defining a relationship includes defining a relationship between two locations upon an assignment of a first variable and an address of a second variable in the program.

35. The method of claim 32, wherein defining a relationship includes defining a relationship between two locations upon an assignment of a first variable and a dereference of a second variable in the program.

36. The method of claim 32, wherein defining a relationship includes defining a relationship between two locations upon an assignment of a dereference of a first variable and a second variable.

37. A method of graphing variables in a program, the method comprising:

displaying a plurality of nodes, wherein each node of the plurality of nodes represent at least one variable;

displaying a plurality of lines, wherein at least one line of the plurality of lines represents a pointer relationship between two nodes; and

emanating a flow line from one node of the plurality of nodes to another node of the plurality of nodes so as to create a relationship between the one node and the another node when an assignment in the program is defined.

38. The method of claim 37, wherein displaying a plurality of lines includes display only one line that emanates from one node of the plurality of nodes.

39. The method of claim 37, wherein emanating a flow line includes emanating only one flow line from one node of the plurality of nodes.

40. A computer readable medium having instructions stored thereon for causing a computer to perform a method of graphing variables in a program, the method comprising:

displaying a plurality of nodes, wherein each node of the plurality of nodes represent at least one variable;

displaying a plurality of lines, wherein at least one line of the plurality of lines represents a pointer relationship between two nodes; and

emanating a flow line from one node of the plurality of nodes to another node of the plurality of nodes so as to associate the one node with the another node

when an assignment in the program is defined.

41. A graph for enhancing pointer analysis, the graph comprising:

a plurality of nodes, wherein at least one node of the plurality of nodes represents at least one variable;

a plurality of lines, wherein at least one line of the plurality of lines represents a pointer relationship between at least two nodes of the plurality of nodes; and

at least one flow line, wherein the at least one flow line represents a label relationship between one node of the plurality of nodes and another node of the plurality of nodes.

42. The graph of claim 41, wherein the variable is adapted to be a pointer type through a process that is selected from a group consisting of declaring through a set of predefined data types, type conversion, and type casting.

43. The graph of claim 41, wherein only one line of the plurality of lines emanates from a node of the plurality of nodes.

44. The graph of claim 41, wherein only one flow line emanates from a node of the plurality of nodes.

45. The graph of claim 41, wherein the at least one flow line emanates from the one node of the plurality of nodes to the another node of the plurality of nodes so as to represent that a label of a location of the one node of the plurality of nodes is a subset of a label of a location of the another node of the plurality of nodes.

46. The graph of claim 41, wherein at least two nodes of the plurality of nodes are selectively unified.

47. A data structure to enhance pointer analysis in a program, wherein the program includes at least one assignment statement of variables, wherein each variable includes a name and a content, the data structure comprising:

a data member location; and

a data member flow to represent at least one label relationship.

48. The data structure of claim 47, wherein the data member location includes:

a data member label, wherein the data member label includes at least one data member symbol that represents a name of a variable; and

a data member content that represents a content of the variable or a unification of at least two variables.

49. The data structure of claim 48, wherein the data member flow stores an address of an instantiation of the data structure if an assignment statement is defined for two variables, and wherein the instantiation is related to one of the two variables.

50. The data structure of claim 49, wherein the data structure includes a method member propagate, wherein the method member propagate causes a propagation of the at least one data member symbol so as to make the data member label of one instantiation of the data structure a subset of a data member label of another instantiation of the data structure.

51. The data structure of claim 49, wherein the data structure includes a method member unify, wherein the method member unify merges a data member label of one instantiation of the data structure with a data member label of another instantiation of the data structure, and wherein the method member unify unifies a data member content of one instantiation of the data structure with a data member content of another instantiation of the data structure.

52. A method for enhancing pointer analysis, the method comprising:

processing a plurality of assignment statements in a program to derive a plurality of sets of information, wherein the plurality of sets of information is distributed among a plurality of levels of indirection; and

unifying selectively sets of information in at least one level of indirection so as to allow a desired level of analytical precision within a desired duration of pointer analysis.

53. The method of claim 52, wherein the act of unifying includes unifying sets of information in all levels of indirection except for a first level of indirection.

54. The method of claim 52, wherein the desired duration of pointer analysis is about linearly proportional to the size of the program.

55. A system for enhancing pointer analysis of a program, wherein the program includes at least one source file, the system comprising:

a compiler to compile the at least one source file to produce an intermediate language;

a builder receptive to the intermediate language to build a tree that represents the at least one source file; and

an analyzer to analyze the tree to produce an object file, wherein the object file contains at least one relationship between two variables in an assignment statement in the program, wherein the relationship defines that a set of symbols relating to one of the two variables is a subset of a set of symbols relating to an other of the two variables.

56. The system of claim 55, further comprising a linker to link a plurality of object files of the program so as to produce a pointer analysis for the program.

METHODS FOR ENHANCING POINTER ANALYSES

Abstract of the Disclosure

Methods are described that enhance pointer analysis for programs. Whereas previous methods are constrained by the extremes of an inverse relationship between time and information, the present methods selectively unify information so as to allow a desired level of analytical decision within a desired duration of analysis. One aspect of the present invention includes selectively retaining information at a first order of indirection based on variables in an assignment statement while unifying information at subsequent orders of indirection. The methods are used for pointer variables, but are equally useful to function definitions, function calls, function pointers, indirect function calls, and others. The methods may be used in client analysis tools such as code browsers and slicing tools.

"Express Mail" mailing label number: EL510314740US
Date of Deposit: January 21 2000
I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231
Printed Name Shawn L. Hise
Signature

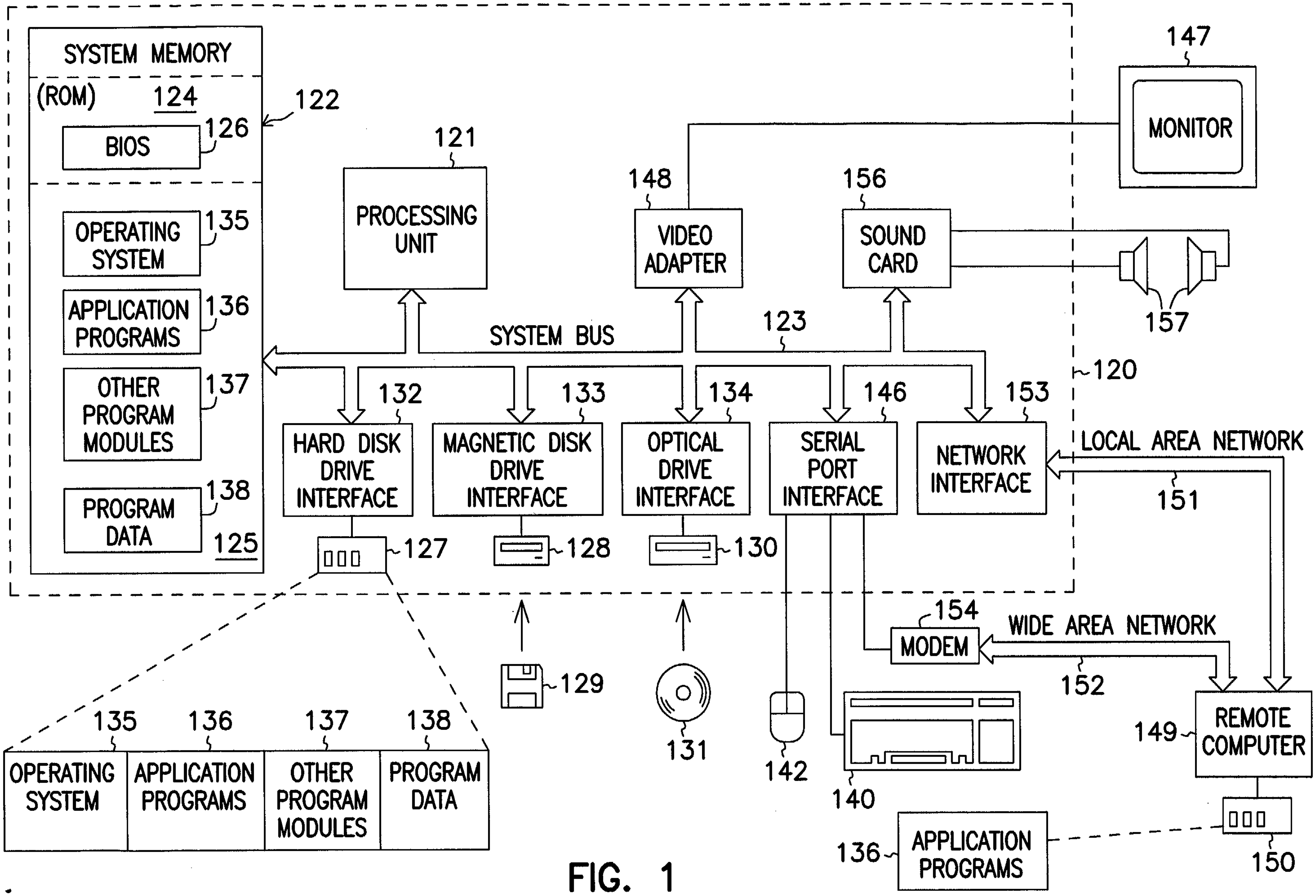

SYSTEM MEMORY
(ROM) 124
BIOS 126
122
OPERATING SYSTEM 135
APPLICATION PROGRAMS 136
OTHER PROGRAM MODULES 137
PROGRAM DATA 138
125
PROCESSING UNIT 121
VIDEO ADAPTER 148
SOUND CARD 156
MONITOR 147
157
SYSTEM BUS
123
120
HARD DISK DRIVE INTERFACE 132
MAGNETIC DISK DRIVE INTERFACE 133
OPTICAL DRIVE INTERFACE 134
SERIAL PORT INTERFACE 146
NETWORK INTERFACE 153
LOCAL AREA NETWORK
151
127
128
130
129
131
142
140
MODEM 154
WIDE AREA NETWORK
152
REMOTE COMPUTER 149
150
APPLICATION PROGRAMS 136
OPERATING SYSTEM 135
APPLICATION PROGRAMS 136
OTHER PROGRAM MODULES 137
PROGRAM DATA 138

FIG. 1

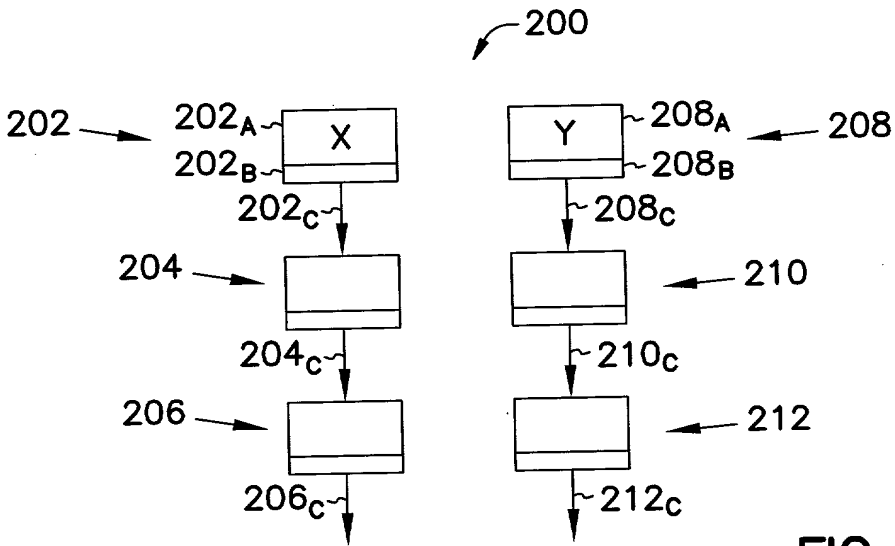


FIG. 2A

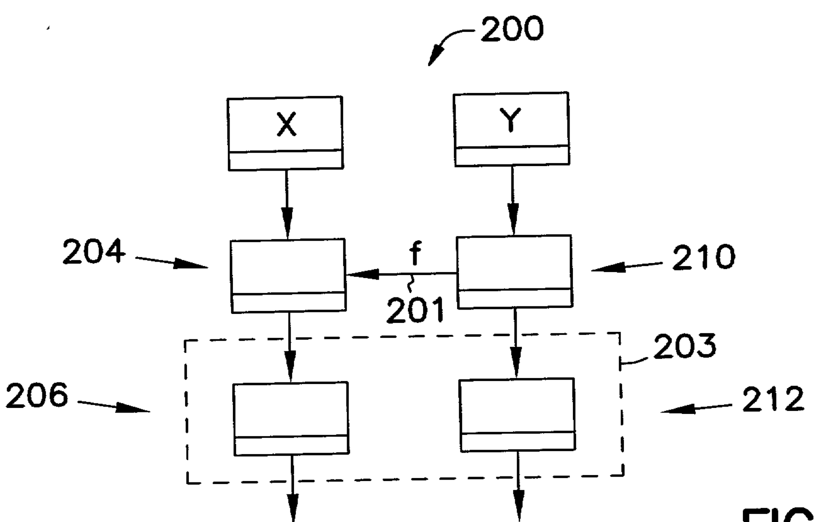


FIG. 2B

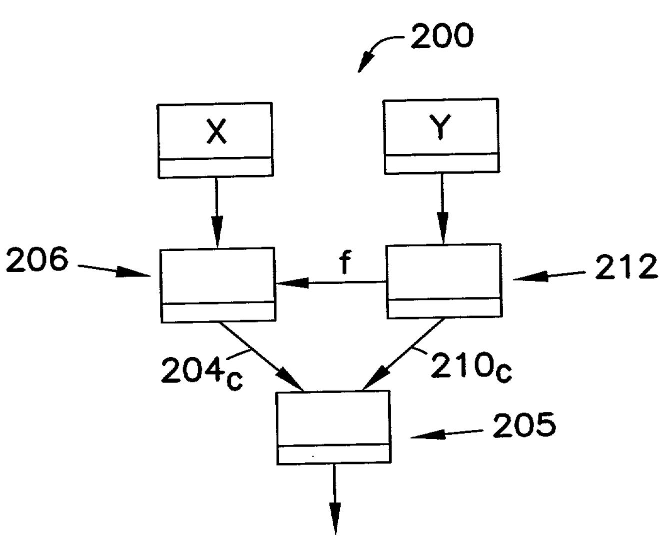


FIG. 2C

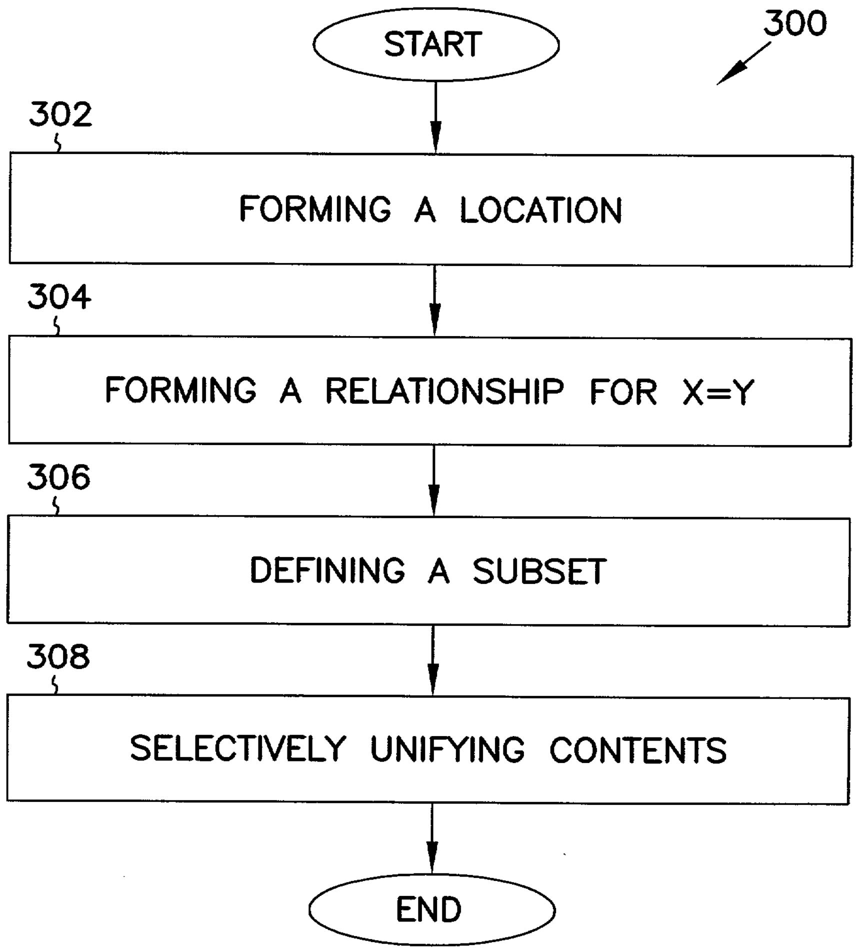

START
300
302
FORMING A LOCATION
304
FORMING A RELATIONSHIP FOR X=Y
306
DEFINING A SUBSET
308
SELECTIVELY UNIFYING CONTENTS
END

FIG. 3

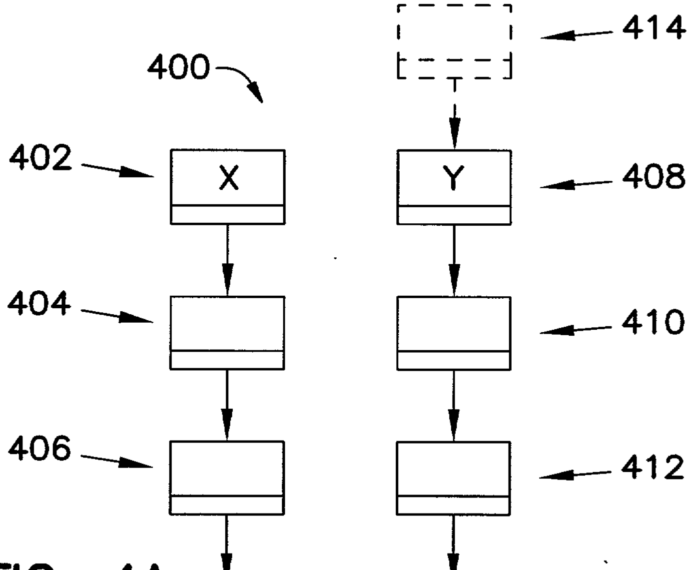


FIG. 4A

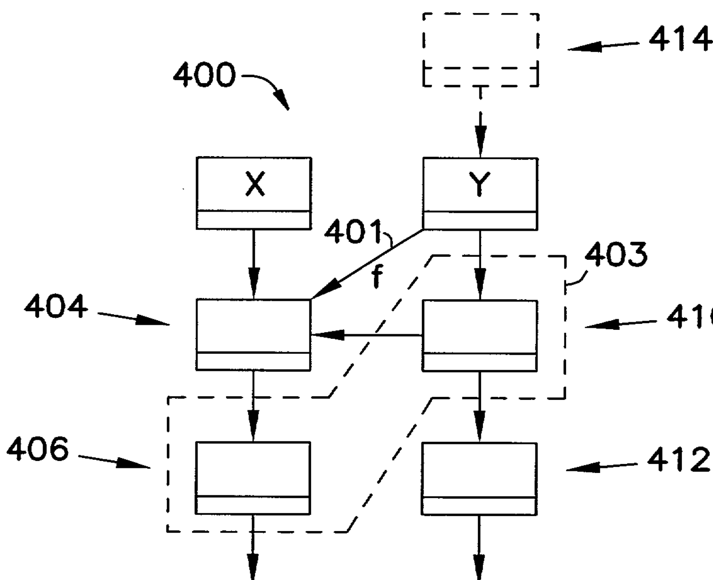


FIG. 4B

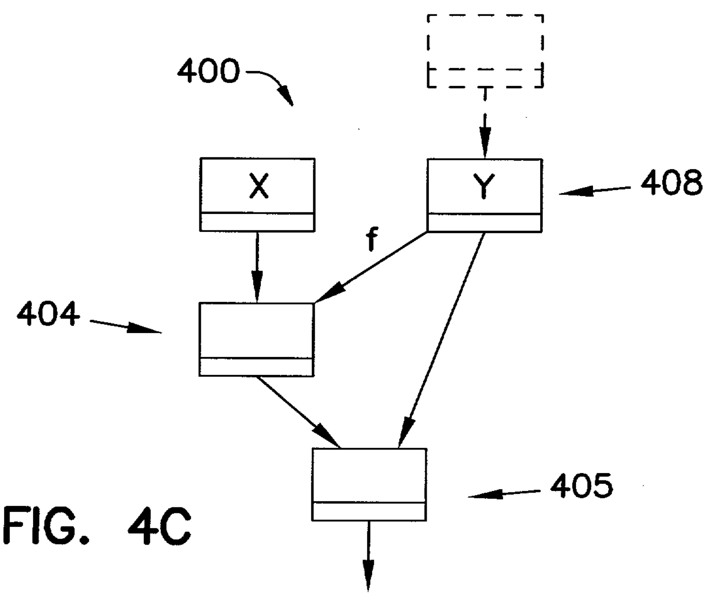


FIG. 4C

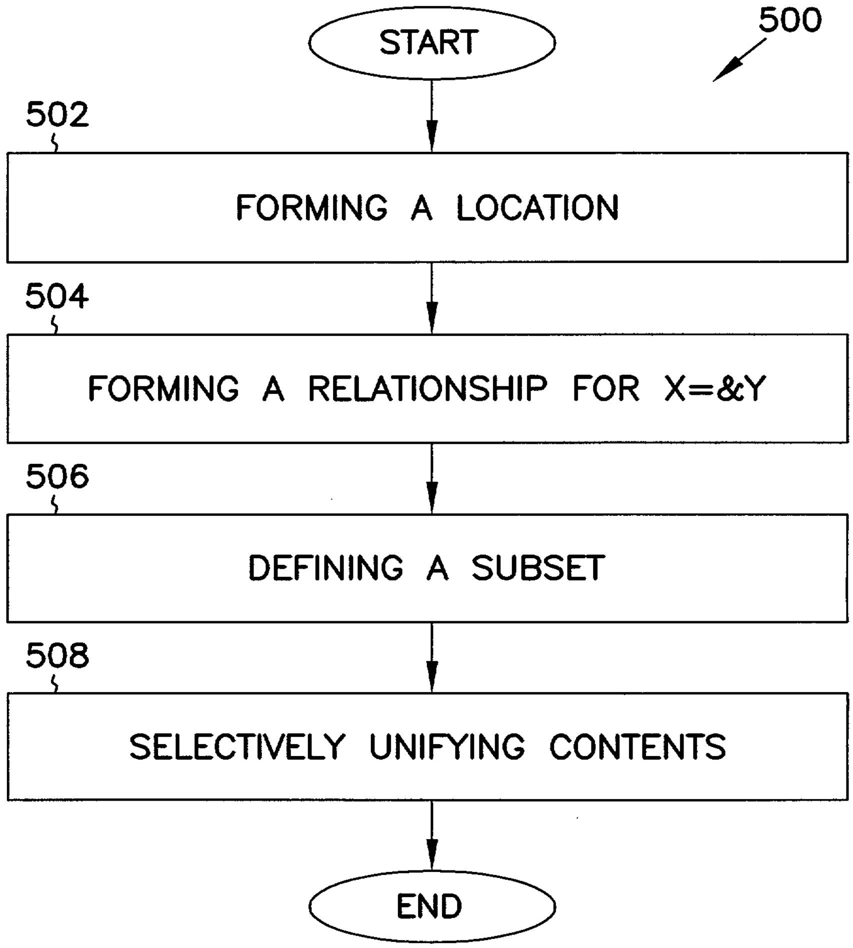

START
500
502
FORMING A LOCATION
504
FORMING A RELATIONSHIP FOR X=&Y
506
DEFINING A SUBSET
508
SELECTIVELY UNIFYING CONTENTS
END

FIG. 5

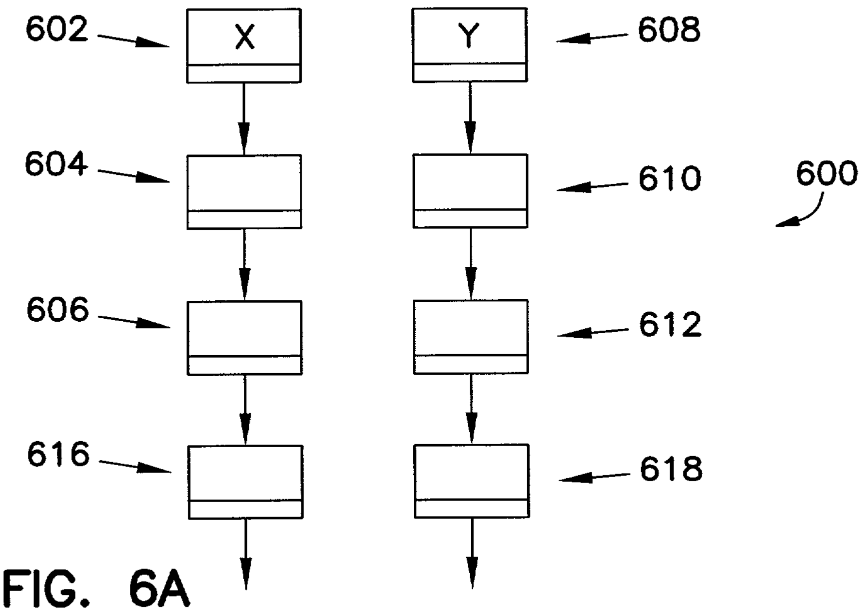


FIG. 6A

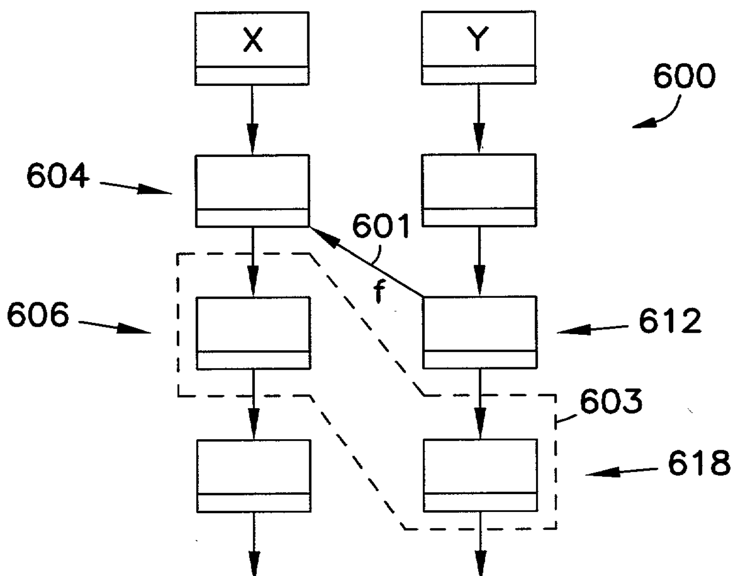


FIG. 6B

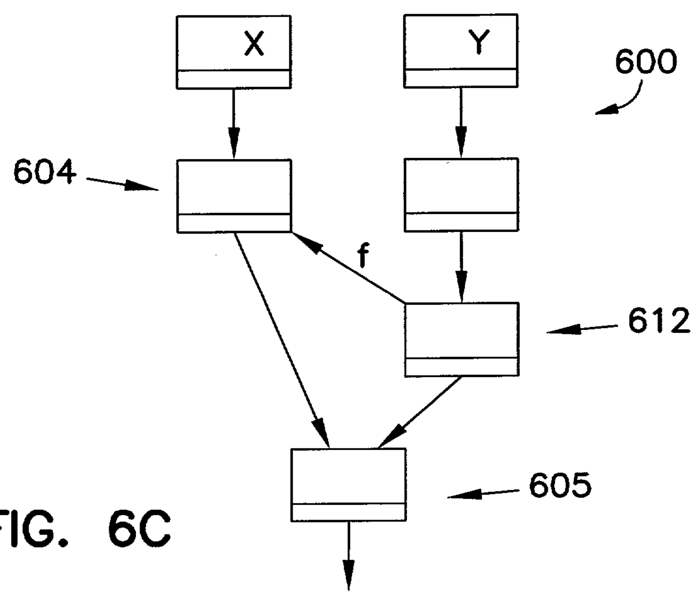


FIG. 6C

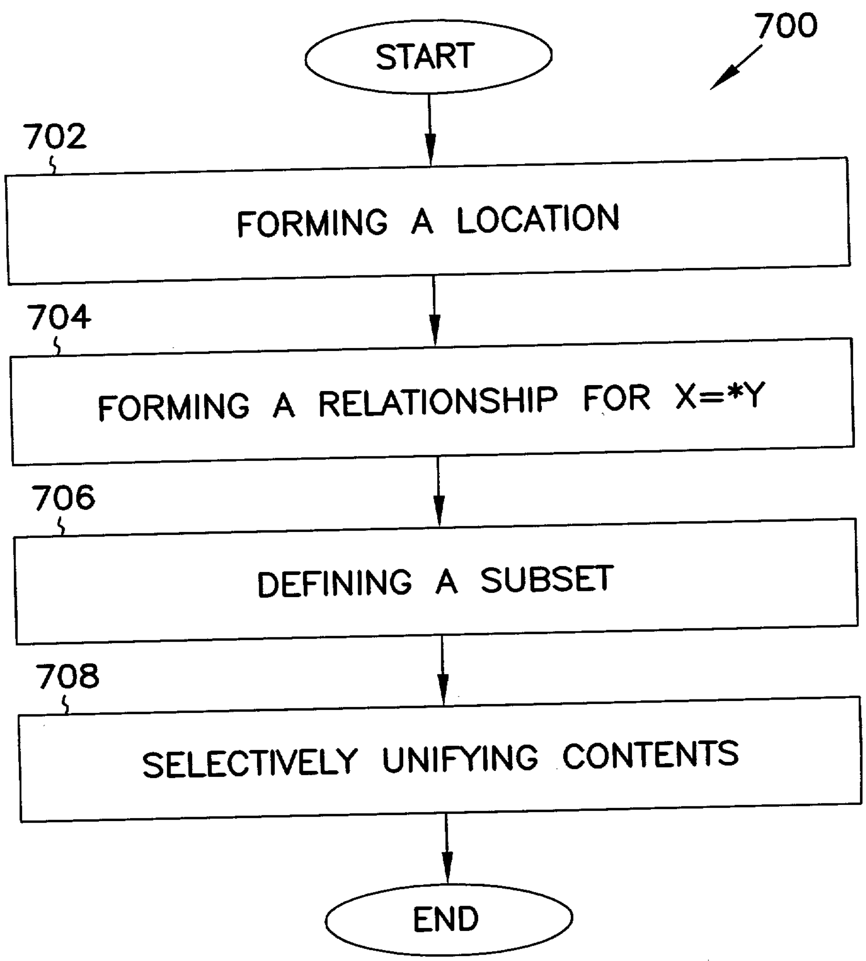

START
700
702
FORMING A LOCATION
704
FORMING A RELATIONSHIP FOR X=*Y
706
DEFINING A SUBSET
708
SELECTIVELY UNIFYING CONTENTS
END

FIG. 7

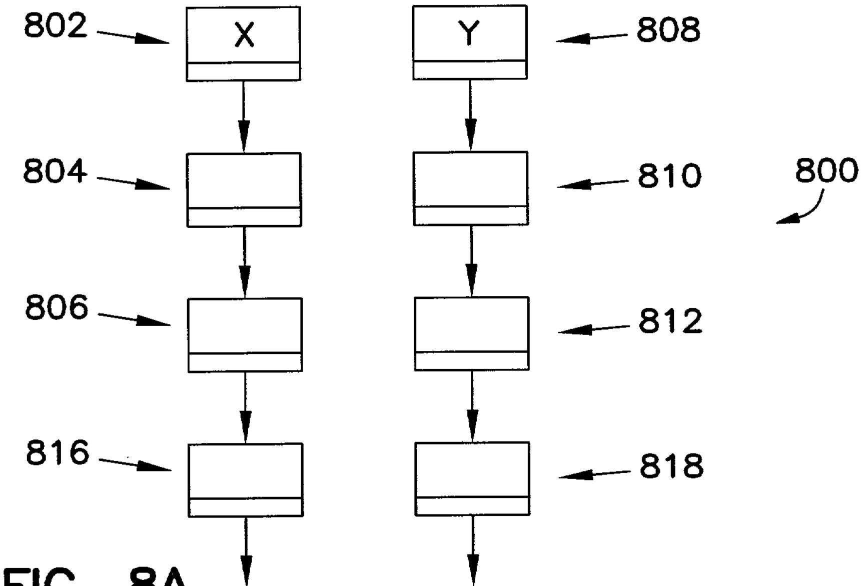


FIG. 8A

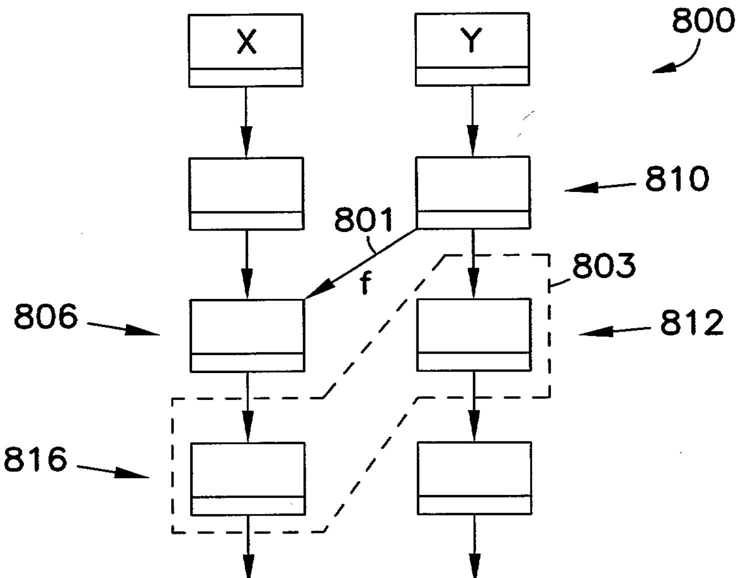


FIG. 8B

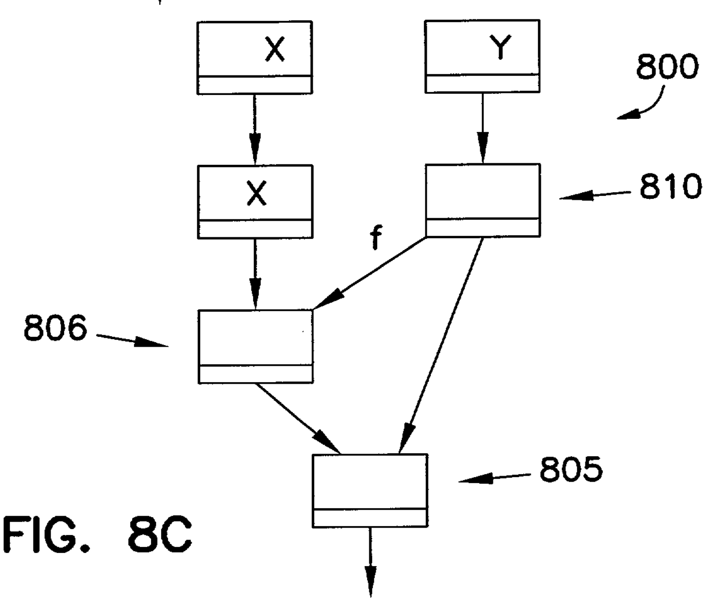
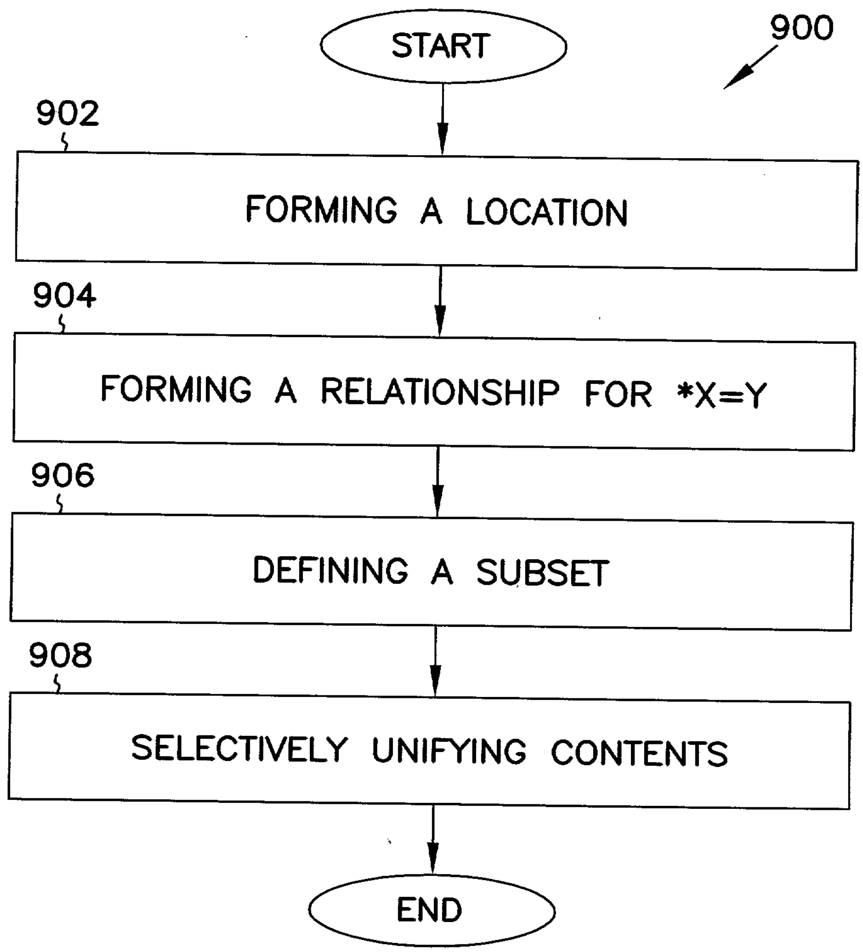


FIG. 8C



900
START
902
FORMING A LOCATION
904
FORMING A RELATIONSHIP FOR *X=Y
906
DEFINING A SUBSET
908
SELECTIVELY UNIFYING CONTENTS
END

FIG. 9

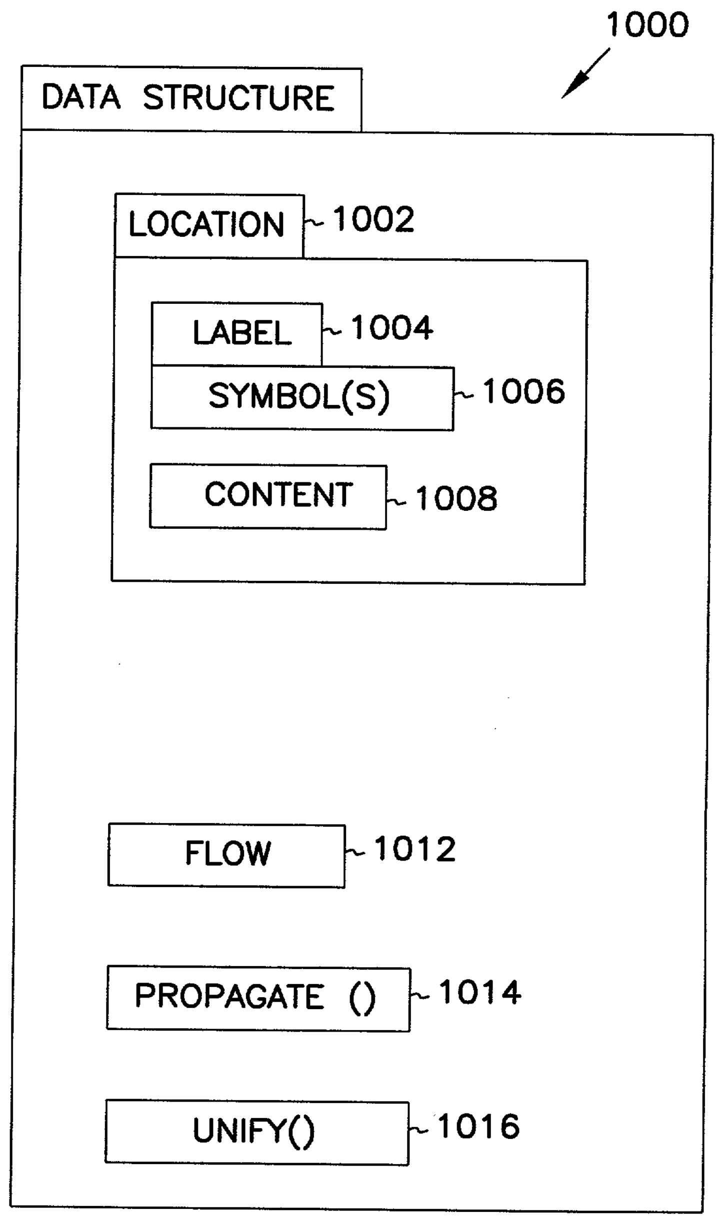
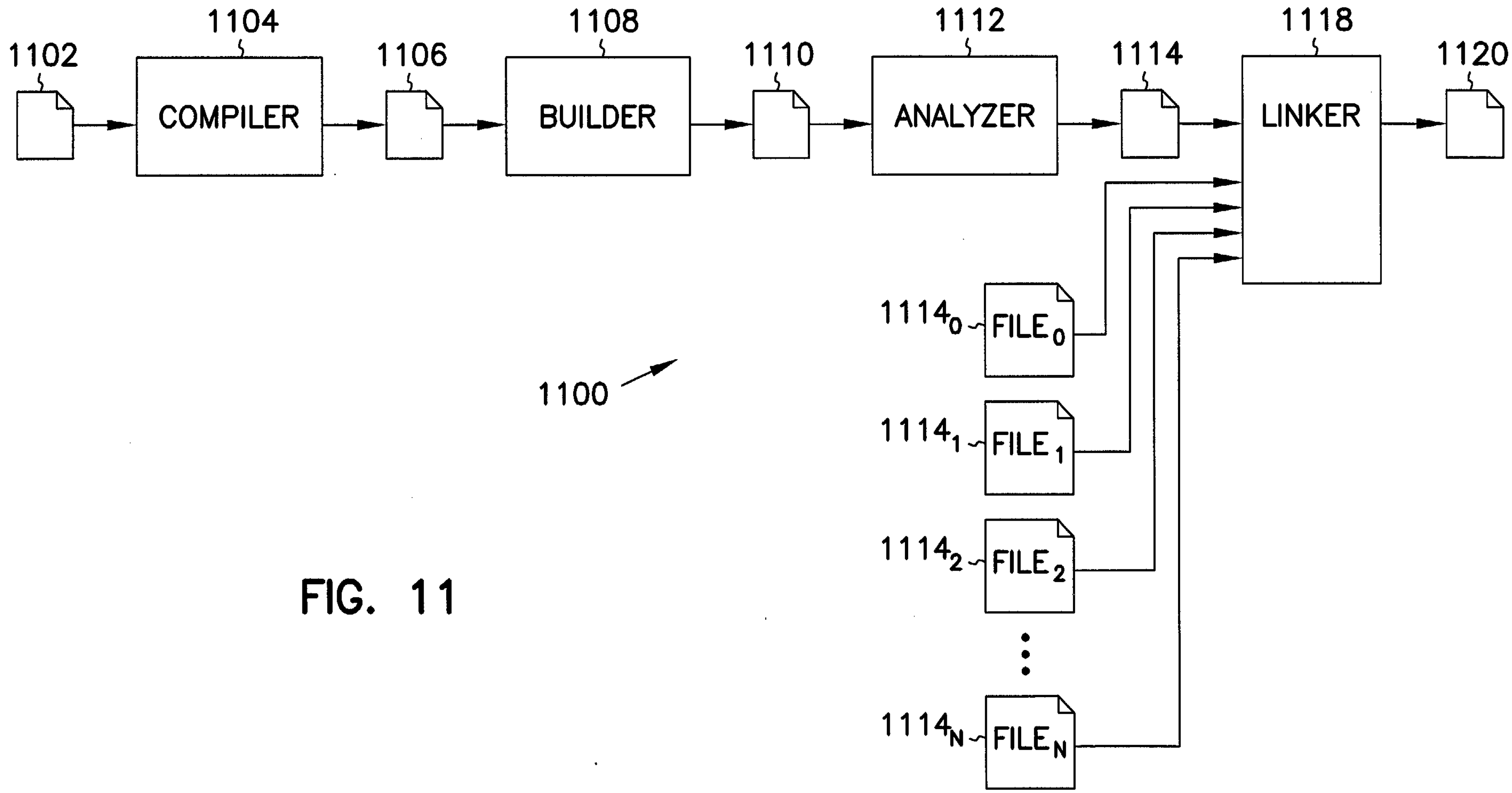

1000
DATA STRUCTURE
LOCATION
1002
LABEL
1004
SYMBOL(S)
1006
CONTENT
1008
FLOW
1012
PROPAGATE ()
1014
UNIFY()
1016

FIG. 10



1102
1104
COMPILER
1106
1108
BUILDER
1110
1112
ANALYZER
1114
1118
LINKER
1120
1100
1114_0
FILE_0
1114_1
FILE_1
1114_2
FILE_2
1114_N
FILE_N

FIG. 11

SCHWEGMAN, LUNDBERG, WOESSNER & KLUTH, P.A.

United States Patent Application

COMBINED DECLARATION AND POWER OF ATTORNEY

As a below named inventor I hereby declare that: my residence, post office address and citizenship are as stated below next to my name; that

I verily believe I am the original, first and sole inventor of the subject matter which is claimed and for which a patent is sought on the invention entitled: METHODS FOR ENHANCING POINTER ANALYSES.

The specification of which is attached hereto.

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the patentability of this application in accordance with 37 C.F.R. § 1.56 (attached hereto). I also acknowledge my duty to disclose all information known to be material to patentability which became available between a filing date of a prior application and the national or PCT international filing date in the event this is a Continuation-In-Part application in accordance with 37 C.F.R. §1.63(e).

I hereby claim foreign priority benefits under 35 U.S.C. §119(a)-(d) or 365(b) of any foreign application(s) for patent or inventor's certificate, or 365(a) of any PCT international application which designated at least one country other than the United States of America, listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on the basis of which priority is claimed:

No such claim for priority is being made at this time.

I hereby claim the benefit under 35 U.S.C. § 119(e) of any United States provisional application(s) listed below:

No such claim for priority is being made at this time.

I hereby claim the benefit under 35 U.S.C. § 120 or 365(c) of any United States and PCT international application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT international application in the manner provided by the first paragraph of 35 U.S.C. § 112, I acknowledge the duty to disclose material information as defined in 37 C.F.R. § 1.56(a) which became available between the filing date of the prior application and the national or PCT international filing date of this application:

No such claim for priority is being made at this time.

I hereby appoint the following attorney(s) and/or patent agent(s) to prosecute this application and to transact all business in the Patent and Trademark Office connected herewith:

Adams, Gregory J.	Reg. No. 44,494
Anglin, J. Michael	Reg. No. 24,916
Bentley, Dwayne L.	Reg. No. P-45,947
Bianchi, Timothy E.	Reg. No. 39,610
Billion, Richard E.	Reg. No. 32,836
Black, David W.	Reg. No. 42,331
Brennan, Leoniede M.	Reg. No. 35,832
Brennan, Thomas F.	Reg. No. 35,075
Brooks, Edward J., III	Reg. No. 40,925
Chu, Dinh C.P.	Reg. No. 41,676
Clark, Barbara J.	Reg. No. 38,107
Crouse, Daniel D.	Reg. No. 32,022
Dahl, John M.	Reg. No. 44,639
Drake, Eduardo E.	Reg. No. 40,594
Eliseeva, Maria M.	Reg. No. 43,328
Embretson, Janet E.	Reg. No. 39,665
Fogg, David N.	Reg. No. 35,138
Fordenbacher, Paul J.	Reg. No. 42,546
Forrest, Bradley A.	Reg. No. 30,837
Harris, Robert J.	Reg. No. 37,346
Huebsch, Joseph C.	Reg. No. 42,673
Jurkovich, Patti J.	Reg. No. 44,813
Kalis, Janal M.	Reg. No. 37,650
Kaufmann, John D.	Reg. No. 24,017
Klima-Silberg, Catherine I.	Reg. No. 40,052
Kluth, Daniel J.	Reg. No. 32,146
Lacy, Rodney L.	Reg. No. 41,136
Leffert, Thomas W.	Reg. No. 40,697
Lemaire, Charles A.	Reg. No. 36,198
Litman, Mark A.	Reg. No. 26,390
Lundberg, Steven W.	Reg. No. 30,568
Mack, Lisa K.	Reg. No. 42,825
Maki, Peter C.	Reg. No. 42,832
Malen, Peter L.	Reg. No. 44,894
Mates, Robert E.	Reg. No. 35,271
McCrackin, Ann M.	Reg. No. 42,858
Nama, Kash	Reg. No. 44,255
Nelson, Albin J.	Reg. No. 28,650
Nielsen, Walter W.	Reg. No. 25,539
Oh, Allen J.	Reg. No. 42,047
Padys, Danny J.	Reg. No. 35,635
Parker, J. Kevin	Reg. No. 33,024
Peacock, Gregg A.	Reg. No. 45,001
Perdok, Monique M.	Reg. No. 42,989
Polglaze, Daniel J.	Reg. No. 39,801
Prout, William F.	Reg. No. 33,995
Sako, Katie E.	Reg. No. 32,628
Schumm, Sherry W.	Reg. No. 39,422
Schwegman, Micheal L.	Reg. No. 25,816
Shaw, Stephen H.	Reg. No. P-45,404
Slifer, Russell D.	Reg. No. 39,838
Smith, Michael G.	Reg. No. P-45,368
Speier, Gary J.	Reg. No. P-45,458
Steffey, Charles E.	Reg. No. 25,179
Terry, Kathleen R.	Reg. No. 31,884
Tong, Viet V.	Reg. No. P-45,416
Viksnins, Ann S.	Reg. No. 37,748
Woessner, Warren D.	Reg. No. 30,440

I hereby authorize them to act and rely on instructions from and communicate directly with the person/assignee/attorney/firm/organization/who/which first sends/sent this case to them and by whom/which I hereby declare that I have consented after full disclosure to be represented unless/until I instruct Schwegman, Lundberg, Woessner & Kluth, P.A. to the contrary.

Please direct all correspondence in this case to Schwegman, Lundberg, Woessner & Kluth, P.A. at the address indicated below:

P.O. Box 2938, Minneapolis, MN 55402
Telephone No. (612)373-6900

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of sole inventor : **Manuvir Das**
Citizenship: **India** Residence: **Kirkland, WA**
Post Office Address: 13384 NE 134th Place
Kirkland, WA 98034

Signature: Manuvir Das MDas Date: 20 JAN 2000

Full Name of inventor:
Citizenship: Residence:
Post Office Address:

Signature: Date:

§ 1.56 Duty to disclose information material to patentability.

(a) A patent by its very nature is affected with a public interest. The public interest is best served, and the most effective patent examination occurs when, at the time an application is being examined, the Office is aware of and evaluates the teachings of all information material to patentability. Each individual associated with the filing and prosecution of a patent application has a duty of candor and good faith in dealing with the Office, which includes a duty to disclose to the Office all information known to that individual to be material to patentability as defined in this section. The duty to disclose information exists with respect to each pending claim until the claim is canceled or withdrawn from consideration, or the application becomes abandoned. Information material to the patentability of a claim that is canceled or withdrawn from consideration need not be submitted if the information is not material to the patentability of any claim remaining under consideration in the application. There is no duty to submit information which is not material to the patentability of any existing claim. The duty to disclose all information known to be material to patentability is deemed to be satisfied if all information known to be material to patentability of any claim issued in a patent was cited by the Office or submitted to the Office in the manner prescribed by §§ 1.97(b)-(d) and 1.98. However, no patent will be granted on an application in connection with which fraud on the Office was practiced or attempted or the duty of disclosure was violated through bad faith or intentional misconduct. The Office encourages applicants to carefully examine:

(1) prior art cited in search reports of a foreign patent office in a counterpart application, and

(2) the closest information over which individuals associated with the filing or prosecution of a patent application believe any pending claim patentably defines, to make sure that any material information contained therein is disclosed to the Office.

(b) Under this section, information is material to patentability when it is not cumulative to information already of record or being made of record in the application, and

(1) It establishes, by itself or in combination with other information, a prima facie case of unpatentability of a claim; or

(2) It refutes, or is inconsistent with, a position the applicant takes in:

(i) Opposing an argument of unpatentability relied on by the Office, or

(ii) Asserting an argument of patentability.

A prima facie case of unpatentability is established when the information compels a conclusion that a claim is unpatentable under the preponderance of evidence, burden-of-proof standard, giving each term in the claim its broadest reasonable construction consistent with the specification, and before any consideration is given to evidence which may be submitted in an attempt to establish a contrary conclusion of patentability.

(c) Individuals associated with the filing or prosecution of a patent application within the meaning of this section are:

(1) Each inventor named in the application:

(2) Each attorney or agent who prepares or prosecutes the application; and

(3) Every other person who is substantively involved in the preparation or prosecution of the application and who is associated with the inventor, with the assignee or with anyone to whom there is an obligation to assign the application.

(d) Individuals other than the attorney, agent or inventor may comply with this section by disclosing information to the attorney, agent, or inventor.